문장 분류기 구현

- 1팀: NLP 이렇게 하는거조 –

신혜인, 구교선, 김동현, 신선미, 유도형, 전은정, 조상현, 하영현

문장 분류기 구현

목차

1. 프로젝트 개요

다양한 BERT 모델의 성능 비교

구현 내용

- 제한된 환경에서 다양한 BERT 모델을 기반으로 주어진 데이터로 학습하여 Sentiment Classification Model을 생성한다.
- 모델 별 성능을 비교하여 Best 모델을 선정 및 WandB를 활용하여 LOSS/ACC를 시각화 한다.

활용 라이브러리 및 프레임워크

- PyTorch
- Hugging Face Transformer
- Bert Model
- AlBert Model
- RoBERTa Model
- WandB (Sweep)

개발 환경

- Colab Pro Plus

1. 프로젝트 개요

프로젝트 기대효과

기업 측면

- 감정 분석을 통해 품질, 서비스 등에 대한 개선 방향 수립 가능.
- 범람하는 품질, 서비스 평가 데이터의 활용 가능성 재고.

학문 측면

- 감정 분석 분야에서 높은 성능을 가지는 모델 판별에 기여.
- Multilingual Model 혹은 KoBert를 활용할 경우 영어 외 다른 언어에 대한 감정 분석 시도 가능성 기대.

2. 팀 구성 및 역할

역할	
신혜인	데일리 미팅일지 작성, bert-large-uncased, bert-base-uncased-yelp-polarity 모델
구교선	baseline 코드 분석 및 버그 수정, bert-large-cased 모델
김동현	baseline 기준 learning rate 변화 loss 측정, 데이터 cleaning & normalization
신선미	baseline 코드 정리 및 모듈화, wandb 이용하여 데이터 기록
유도형	baseline 코드 분석 및 버그 수정, baseline 코드 정리 및 모듈화
전은정	model best setting 탐색, wandb 이용하여 데이터 기록
조상현	wandb 데이터 기록, Albert 모델, 결과데이터 정리 및 발표
하영현	wandb 기본 설정, bert-base-uncased, roberta-base 모델

주요 업무

베이스라인 코드 수정 및 작성

- baseline 코드 분석
- 요소별 가설 수립, 코드 실행 후 제출

탐색적 분석 및 전처리

- 짧은 단어, 불용어, 어간추출

모델 별 성능 산출

- 모델 별 best setting 을 바탕으로 epoch, batch size, learning rate 설정

결과 비교 및 보고서 작성

- 선정 성능 지표 기준으로 결과 비교
- 비교 결과와 원인 보고서 작성

3.1 단계 별 프로세스

제한된 환경에서 다양한 모델 비교 및 최적의 모델 선정

베이스라인 버그 탐색	후보 모델 선정	성능 비교	평가 및 보완점 정의
베이스라인 코드 내 버그 수정으로 성능 향상	학습 및 실습한 모델 중에 선정	모델별 최종 성능 비교	프로젝트 수행 과정 자체평가 진행 및 보완점 정의
베이스라인 버그 수정 테스트용 collate function 에서 sorted indice code 삭제 **베이스라인 코드 재작성** 제공된 베이스라인 코드 기반 프로젝트 목적에 부합하는 베이스라인 코드 재작성	**후보 모델 정의** BERT Model AlBERT Model RoBERTa Model **모델별 기본 hyper-parameter 탐색** 논문 및 documentation에 제공되는 기본 파라미터 탐색	**성능 지표 선정** Kaggle score Train score Valid Score 모델 구동 시간 **Best Model 선정** 모델 별 성능 지표 값을 비교해 선정	**자체평가** **제한사항 정의** 프로젝트 수행간에 발생한 제한사항 정의 **보완점 정의** 모델들의 학습 실패 원인 추론

3.2 전체 일정

전체 기간 : 7. 13 ~ 7.19

총 일수 : 7 일

3.3 모델 별 구성요소 선정

논문 및 Documentation에서 제안하는 setting 값을 best setting 값으로 정의

모델 별 best setting 탐색

Model	epoch	batch_size	Lr
BERT-base-cased	4	32	5e-05
BERT-base-uncased	4	32	5e-05
RoBERTa-base	3	64	3e-05
ALBERT-base-v2	5	32	2e-05
Bert-base-uncased-yelp-polarity	5	16	5e-05

4.1 베이스라인 디버깅

- 베이스라인 코드 구동 방식 정리 및 오류 탐색
- Preprocess, Train/Test 로 나눠 팀원 개별 스터디 및 디버깅 시행
- collate_fn_style, collate_fn_style_test 에서 attention_mask 위치 및 sorted_index를 수정해 디버깅
- test acc 0.513 -> 0.982

1. Preprocess

```
def make_id_file(task, tokenizer):
    def make_data_strings(file_name):
        data_strings = []
        with open(os.path.join(file_name), 'r', encoding='utf-8') as f:
            id_file_data = [tokenizer.encode(line.lower()) for line in f.readli
        for item in id_file_data:
            data_strings.append(' '.join([str(k) for k in item]))
        return data_strings

    print('it will take some times...')
    train_pos = make_data_strings('/sentiment.train.1')
    train_neg = make_data_strings('/sentiment.train.0')
    dev_pos = make_data_strings('/sentiment.dev.1')
    dev_neg = make_data_strings('/sentiment.dev.0')

    print('make id file finished!')
    return train_pos, train_neg, dev_pos, dev_neg
```

make_id_files 함수는 각 파일을 연 다음, 각 파일에 있는 문장들을 tokenizer를 통해 encode해주고, 결과를 4개의 list로 반환해준다. 이때 대소문자를 구별하지 않는다.

- task: 함수 내에서 단 한 번도 활용되지 않은 파라미터이다. '어떤 목적으로 사용할지' 나타내는 거로 추측할 수는 있으나, 함수 파라미터로 왜 존재하는지 이유를 알 수 없다.
- tokenizer: make_id_files에서 문서를 tokenize할 때 쓸 tokenizer이다.

```
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
```

여기서 tokenizer는 BERT-base-uncased의 tokenizer를 사용할 것이다.

```
from google.colab import files
uploaded = files.upload()
```

Google colab에서 파일을 올리기 위한 목적의 코드이다.

4.1 베이스라인 디버깅

통일된 베이스라인 구성

- 주요 파라미터 설정 값에 따라 별도 모델을 돌릴 수 있는 공통 코드 구성

```
used_model = 'bert-base-uncased' # 적용 모델
cased = False if 'uncased' in used_model else True

train_batch_size = 32
eval_batch_size = 32
test_batch_size = 32

learning_rate = 2e-5
train_epoch = 3
weight_decay = 0.001

wandb_project = "final_project1" # WandB 프로젝트 이름
wandb_team = "goorm-project-nlp-team-1" # WandB 팀명
```

4.2 탐색적 분석 및 전처리

dataset size

데이터 셋	구분	데이터 수	비율
train	positive	266,041	60.01%
	negative	177,218	39.98%
	total	443,259	100%
dev	positive	2,000	50.00%
	negative	2,000	50.00%
	total	4,000	100%

content

```
['i was sadly mistaken .\n',
 'so on to the hoagies , the italian is general run of the mill .\n',
 'minimal meat and a ton of shredded lettuce .\n',
 'nothing really special & not worthy of the $ _num_ price tag .\n',
 'second , the steak hoagie , it is atrocious .\n',
 'i had to pay $ _num_ to add cheese to the hoagie .\n',
 'she told me there was a charge for the dressing on the side .\n',
 'are you kidding me ?\n',
 'i was not going to pay for the dressing on the side .\n',
 'i ordered it without lettuce , tomato , onions , or dressing .\n']
```

4.2 탐색적 분석 및 전처리

짧은 단어 삭제

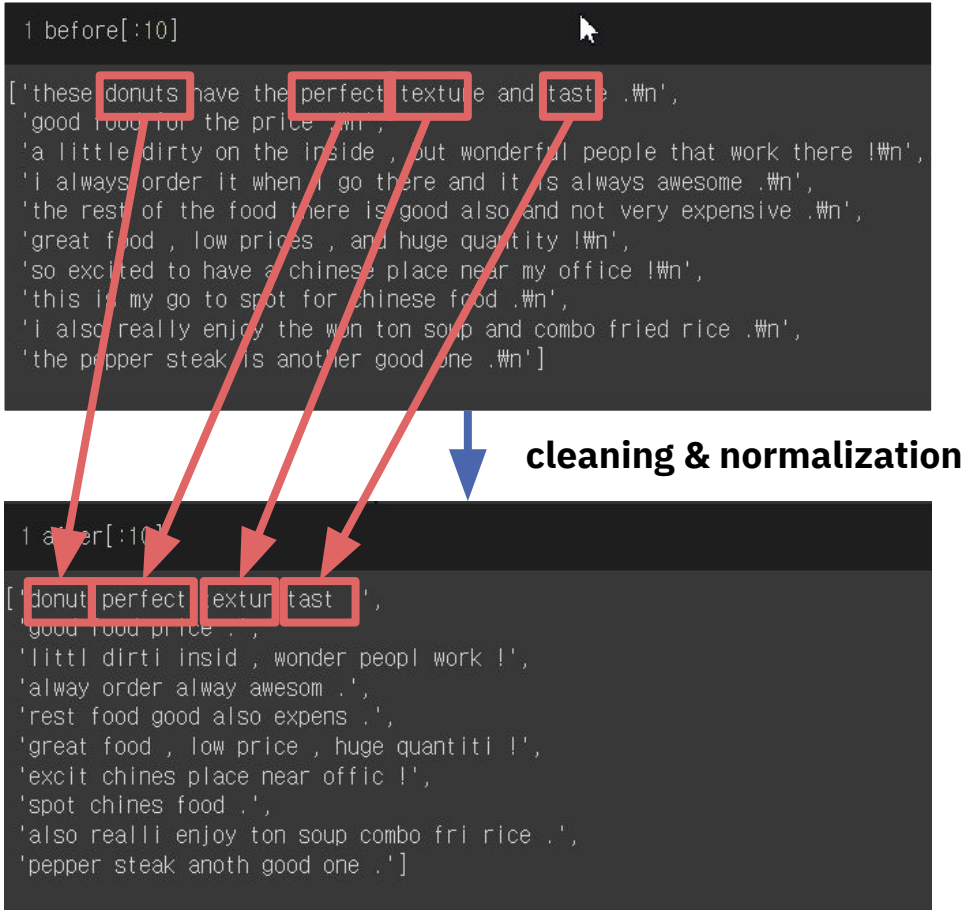
영어로 된 훈련 데이터의 경우
‘길이가 짧은 단어’를 제거 했을 때
‘큰 의미가 없는 단어들’이
제거되는 효과를 볼 수 있다

불용어 삭제

기존의 불용어 데이터 셋과 비교해서
‘의미 없이 용량을 차지하는 단어들'
을 제거해 유의미한 단어들을 선별한다

어간 추출

‘같은 의미를 가진 다른 단어’의 경우
‘하나의 단어’로 일반화 하여
‘문서 내 단어 수’를 줄인다



4.3 모델 탐색

선택 모델 개요

모델명	특징
BERT-base-uncased	모든 layer에서 bidirectional 방식으로 학습, MLM(Masked Language Model)을 수행하여 문맥 파악, NSP(Next Sentence Prediction) 사용, 대소문자 구별하지 않음.
BERT-base-cased	Bert-base-uncased와 동일한 특징. 차이점은 accent 및 대소문자 구분, NER(Named Entity Recognition), POS(Part-of-Speech tagging)가 중요한 task에 적합한 모델.
RoBERTa-base	Bert와는 달리 NSP를 없애고 MLM만으로 pre-training. 모든 input들의 token수가 512에 근접.
ALBERT-base-v2	Bert의 사이즈를 줄인 버전. 문장의 순서를 바꾸는 방식(SOS)으로 학습.
Bert-base-uncased -yelp-polarity	TextAttack의 yelp polarity 데이터를 학습한 데이터로, adversarial attack 방법으로 긍정/부정 데이터를 변환하여 데이터를 생성하고 학습한 모델

4.4 결과 비교

모델 별 성능 비교

Eval metric / Model	Runtime	train loss	train acc	val loss	val acc	kaggle
BERT-base-uncased	1:38:59	**0.02809**	**0.9907**	0.07648	0.978	**0.985**
BERT-base-cased	1:41:39	0.03701	0.9871	0.07102	0.9745	0.973
RoBERTa-base	**0:30:42**	0.06046	0.9787	0.07457	0.9727	0.977
ALBERT-base-v2	2:09:20	0.02832	0.9901	**0.06332**	**0.9805**	0.984

Best model → **BERT-base-uncased**

train loss, train accuracy, kaggle score
총 3개의 지표에서 가장 좋은 성능을 보임

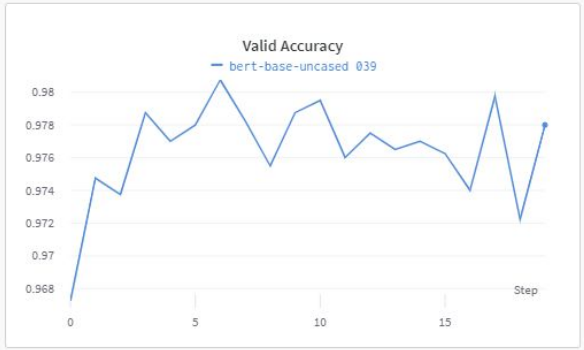
Other model

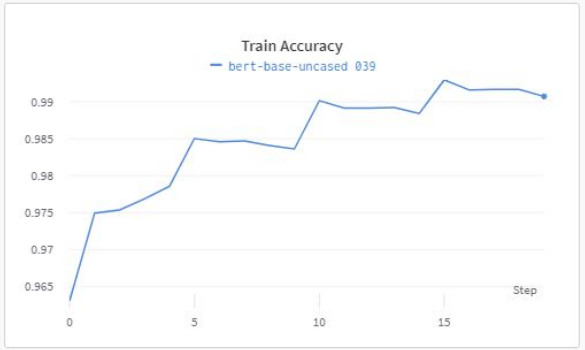
RoBERTa: 가장 짧은 run-time 임에도 Best Model Kaggle score 에 근접

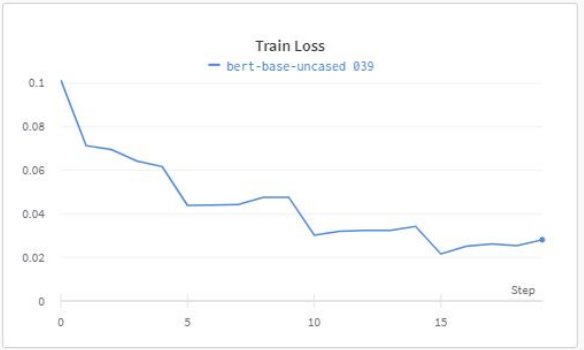
ALBERT: validation accuracy가 가장 높지만 kaggle score가 미세하게 낮음

4.4 결과 비교

"BERT-base-uncased"



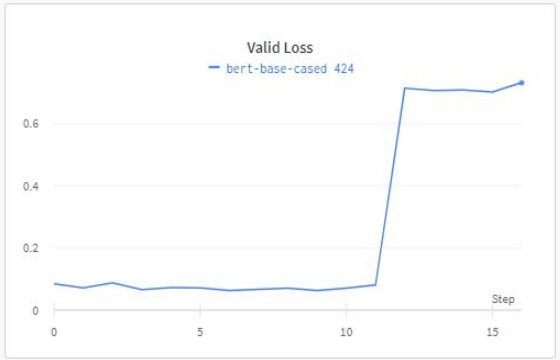


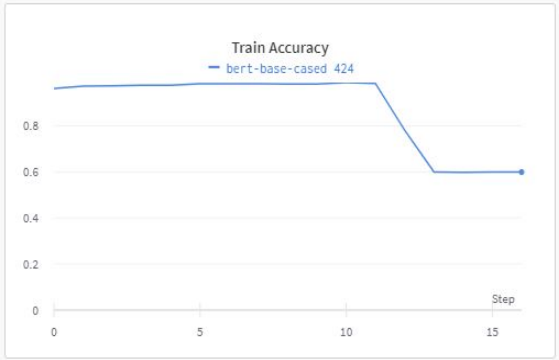


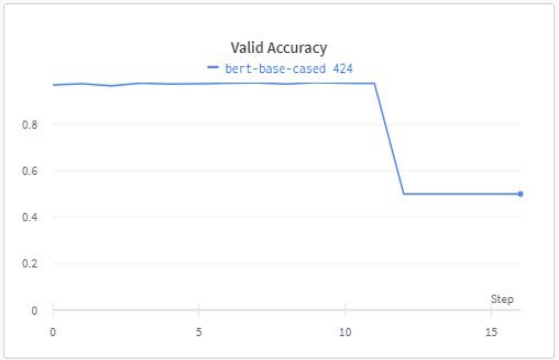
- epoch=4, batch_size=32, lr=5e-05
- Best Model
- train loss, train accuracy, kaggle score 에서 가장 좋은 성능을 보임

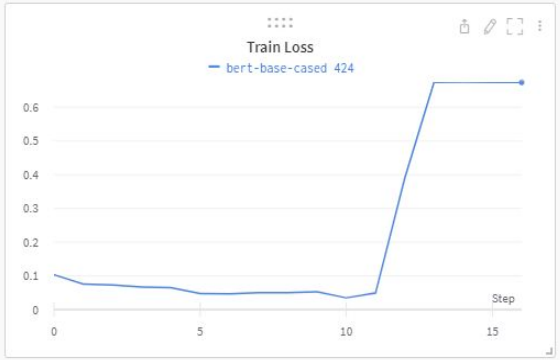
4.4 결과 비교

“BERT-base-cased”





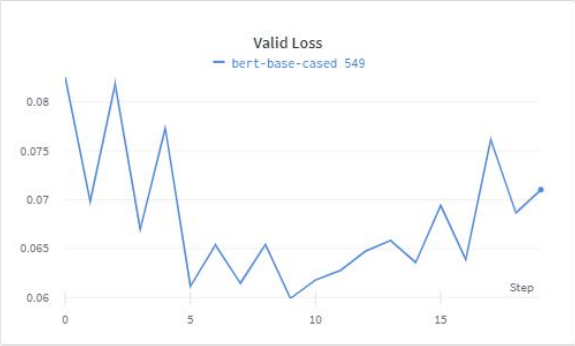


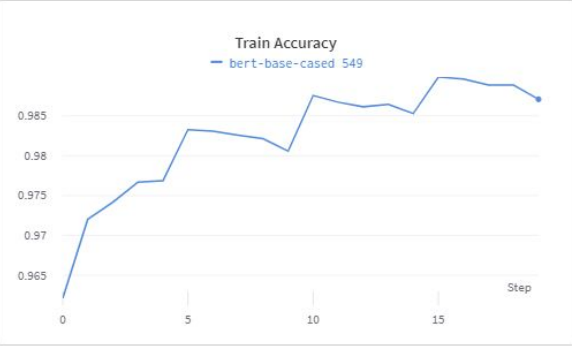


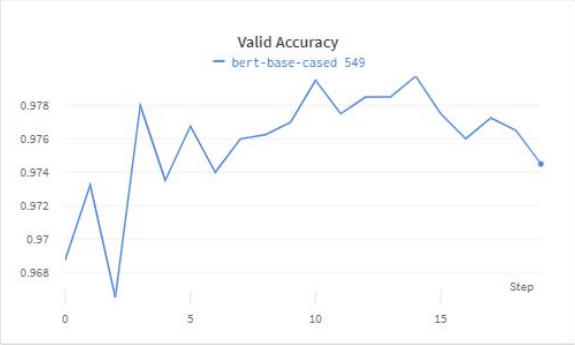
- epoch=4, batch_size=32, lr=5e-05
- weight decay = 0.001
- 3 epoch에서 학습이 제대로 이루어지지 않음 (vanishing/exploding gradient problem)
- kaggle score 0.5로 테스트 실패

4.4 결과 비교

“BERT-base-cased”



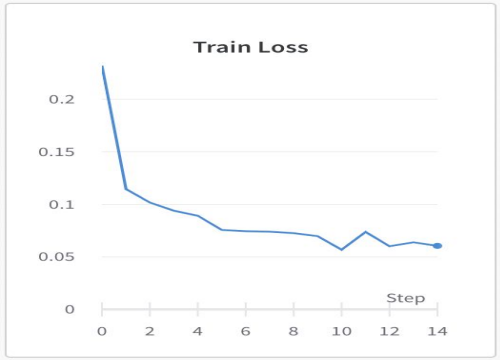


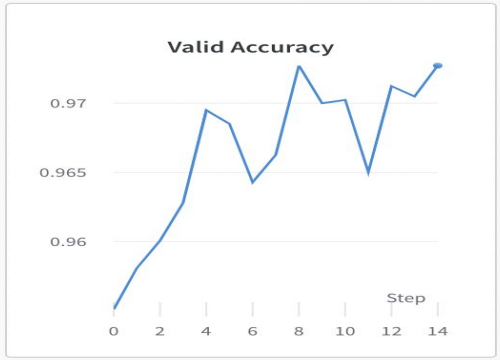


- epoch=4, batch_size=32, lr=5e-05
- weight decay = 0.1
- weight decay 수정으로 성능 개선 확인

4.4 결과 비교

"RoBERTa-base"

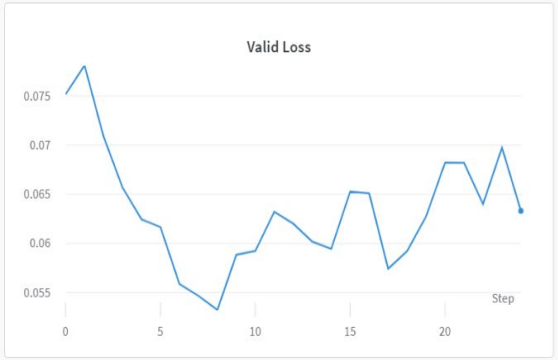


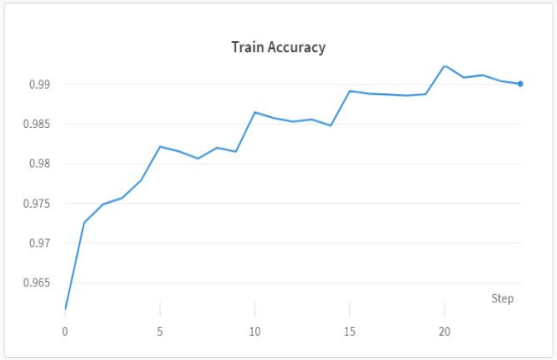


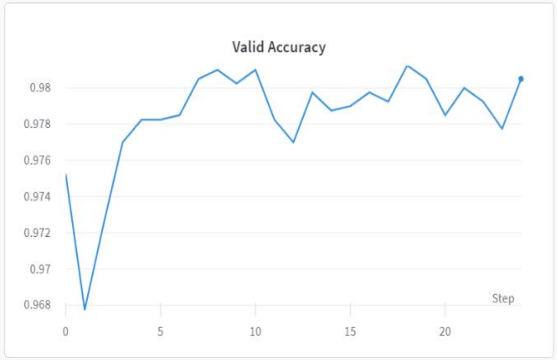
- epoch=3, batch_size=64, lr=3e-05
- 다른 모델에 비해 짧은 시간 내 학습 가능
- 시간에 비해 높은 정확도

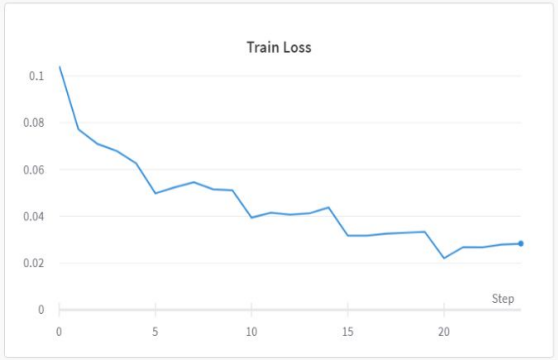
4.4 결과 비교

“ALBERT-base-v2”









- epoch=5, batch_size=32, lr=2e-05
- 다른 모델과 비교하여 학습시간이 긴 편
- 대체적으로 훌륭한 성능을 보임

5.1 자체 평가 및 보완사항

Pros

- BERT를 기반으로 한 다양한 모델 간 비교 목적 달성
- 영어 감성 분석 면에서 효율적인 시간 내에 높은 정확도를 가진 모델 도출
- 다양한 Pre-trained 모델에 대한 이해도 향상

Cons

- 한정된 자원으로 인해 시간 배분 관리의 아쉬움
- sweep을 활용하여 하이퍼 파라미터 튜닝을 통한 모델 성능 향상 가능
- 다양한 모델을 비교하는 것에 초점을 맞춰 각 모델별 특성에 대한 이해 부족

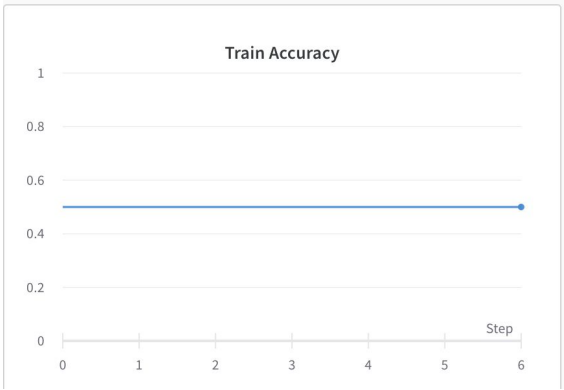
5.2 모델 별 보완 사항

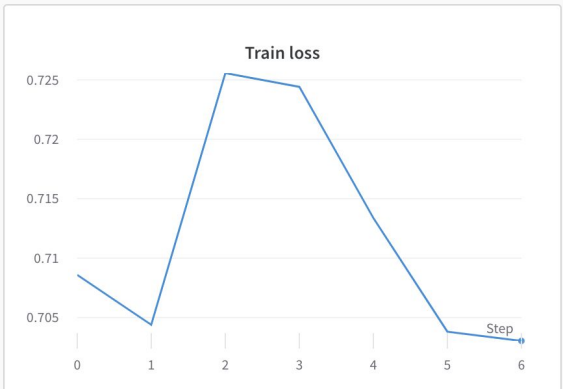
보완 필요 모델

Model \ Eval metric	Runtime	train loss	train acc	val loss	val acc	kaggle
Bert-base-uncased -yelp-polarity	2:50:21	0.6749	0.6002	0.7033	0.5	0.5
BERT-Large-uncased	3:10:08	0.603	0.5000	-	-	0.5
BERT-Large-cased	3:11:58	0.678	0.5000	-	-	0.5

5.2 모델 별 보완 사항

“Bert-Large-cased/uncased”

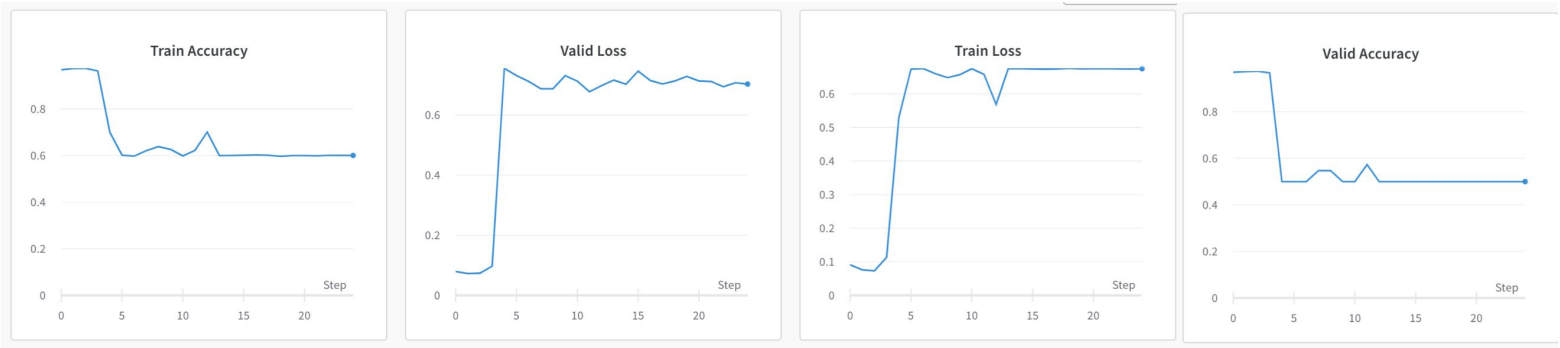




- epoch=3, batch_size=32, lr=5e-05
- 모든 test sentences에 대해 1로 추론
- 원인: 모델 크기에 비해 너무 작은 epoch(3)을 설정하여 학습이 거의 이루어지지 않음
- 모델의 크기가 매우 크기 때문에 여러 번 학습을 실행할 수 없어 개선하지 않고 종료

5.2 모델 별 보완 사항

"bert-base-uncased-yelp-polarity"



- epoch=5, batch_size=16, lr=5e-05
- yelp-polarity 데이터셋 학습 모델로 높은 성능을 기대했으나, 모든 추론 결과가 1로 나와 학습 실패
- 시간 부족으로 학습 실패 상황으로 마무리

6. 참고 문헌

- BERT github https://github.com/google-research/bert
- ALBERT github https://github.com/google-research/albert
- Textattack bert-base-uncased-yelp-polarity https://huggingface.co/textattack/bert-base-uncased-yelp-polarity, https://textattack.readthedocs.io/en/latest/1start/what_is_an_adversarial_attack.html
- Wandb https://docs.wandb.ai/quickstart
- BERT papers with code https://paperswithcode.com/method/bert
- PyTorch https://pytorch.org/docs/stable/index.html
- Jiakai Wei. Forget the Learning Rate, Decay Loss, 2019. arXiv:1905.00094 https://arxiv.org/abs/1905.00094